IDA DOCUMENT D-1243

FY92 CONTRIBUTIONS TO THE OPERATING SYSTEMS STANDARDS WORKING GROUP OF THE NAVY NEXT GENERATION COMPUTER RESOURCES PROGRAM

Karen D. Gordon
Terry Mayfield, *Task Leader*

October 1992

Prepared for
Space and Naval Warfare Systems Command
Ada Joint Program Office

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

IDA Log No. HQ 92-042852

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Group Reports

Group Reports record the findings and results of IDA established working groups and panels composed of senior individuals addressing major issues which otherwise would be the subject of an IDA Report. IDA Group Reports are reviewed by the senior individuals responsible for the project and others as selected by IDA to ensure their high quality and relevance to the problems studied, and are released by the President of IDA.

Papers

Papers, also authoritative and carefully considered products of IDA, address studies that are narrower in scope than those covered in Reports. IDA Papers are reviewed to ensure that they meet the high standards expected of refereed papers in professional journals or formal Agency reports.

Documents

IDA Documents are used for the convenience of the sponsors or the analysis (a) to record substantive work done in quick reaction studies, (b) to record the proceedings of conferences and meetings, (c) to make available preliminary and tentative results of analyses, (d) to record data developed in the course of an investigation, or (e) to forward information that is essentially unanalyzed and unevaluated. The review of IDA Documents is suited to their content and intended use.

The work reported in this document was conducted under contract MDA 903 89 C 0003 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1992	Final

4. TITLE AND SUBTITLE
FY92 Contributions to the Operating Systems Standards Working Group of the Navy Next Generation Computer Resources Program

5. FUNDING NUMBERS
MDA 903 89 C 0003

Task T-D5-725

6. AUTHOR(S)
Karen D. Gordon

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Defense Analyses (IDA)
1801 N. Beauregard St.
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Document D-1243

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Space and Naval Warfare Systems Command
SPAWAR 231-2B3
Washington, D.C. 20363-5100

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, unlimited distribution: December 15, 1992.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (Maximum 200 words)

The Next Generation Computer Resources (NGCR) Program is a Navy standardization effort designed to fulfill the Navy's needs for standard computer resources while at the same time allowing it to take advantage of commercial products and investments and to field new technology more quickly and effectively. The program is centered around the selection and adoption of open system standards in several areas, including backplanes, local area networks, operating systems, project support environments, graphics, and database management systems. IDA is providing support to the NGCR through participation in the NGCR Operating Systems Standards Working Group (OSSWG), a group chartered to identify and help define commercially based operating system interface standards for use in Navy mission-critical computer systems in the mid-1990s and beyond. Because of the results of an extensive and rigorous evaluation process, the NGCR operating system interface standards will be based on POSIX, a family of standards being defined by IEEE Project 1003. This document is a compilation of IDA written contributions to the NGCR OSSWG for fiscal year 1992. It includes copies of IDA contributions to NGCR OSSWG documents and a tabular summary of POSIX real-time application environment profiles.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Open System; POSIX; Real-Time Operating System; Interface Standards; Interoperability.	50
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

IDA DOCUMENT D-1243

FY92 CONTRIBUTIONS TO THE OPERATING SYSTEMS STANDARDS WORKING GROUP OF THE NAVY NEXT GENERATION COMPUTER RESOURCES PROGRAM

Karen D. Gordon
Terry Mayfield, *Task Leader*

October 1992

INSTITUTE FOR DEFENSE ANALYSES
Contract MDA 903 89 C 0003
Task T-D5-725

PREFACE

The Space and Naval Warfare Systems Command (SPAWAR) and the Ada Joint Program Office (AJPO), Office of the Deputy Director of Defense Research and Engineering (Science and Technology), tasked the Institute for Defense Analyses (IDA) to support the Next Generation Computer Resources (NGCR) Operating Systems Standards Working Group (OSSWG) in identifying and helping to define operating system interface standards that can meet Navy mission-critical computing requirements. This document is a compilation of written contributions made by IDA under the task for fiscal year 1992. It is submitted in partial fulfillment of the NGCR Operating System Standards task.

This document was reviewed by the following members of IDA management: Dr. Richard J. Ivanetich and Mr. Terry Mayfield.

CONTENTS

INTRODUCTION

The U.S. Navy is conducting a standardization effort known as the Next Generation Computer Resources (NGCR) Program. The NGCR Program is designed to fulfill the Navy's needs for standard computer resources while at the same time allowing it to take advantage of commercial products and investments and to field new technology more quickly and effectively. The program is centered around the selection and adoption of open system standards in several areas, including backplanes, local area networks, operating systems, project support environments, graphics, and database management systems.

Since January 1989, the Institute for Defense Analyses (IDA) has been providing support to the NGCR Program in the area of operating systems. At that time, the U.S. Navy Space and Naval Warfare Systems Command formed the NGCR Operating Systems Standards Working Group (OSSWG), of which IDA is a member. The OSSWG is chartered to identify and help define commercially based operating system interface standards for use in Navy mission-critical computing systems in the mid-1990s and beyond.

In April 1990, following an extensive and rigorous evaluation process, the NGCR OSSWG selected POSIX, the Portable Operating System Interface for Computer Environments—a family of standards being defined by IEEE Project 1003—as the baseline upon which to establish the NGCR operating system interface standards. Since then, the NGCR OSSWG has been actively participating in the POSIX standardization process. The goal of the NGCR OSSWG participation is to ensure that the POSIX standards evolve in ways that will enable them to better meet Navy mission-critical computing needs. To this end, the NGCR OSSWG continues to re-examine Navy requirements and to seek ways of extending the POSIX standards in directions important to mission-critical computing.

This document contains IDA's contributions to NGCR OSSWG documents and a tabular summary of POSIX real-time application environment profiles.

PART 1

Contributions to

A Next Generation Computer Resources (NGCR) Open Systems Architecture

The draft document referred to above puts forward a vision of an NGCR program-wide open system architecture. As stated at the beginning of the document, its purpose is "to provide clarity and understanding of how the Next Generation Computer Resource (NGCR) interface standards interconnect, interoperate and inter-relate." The 3 December 1991 draft of the document was prepared by a group of NGCR OSSWG members and presented at the 10-12 December 1991 meeting of the NGCR OSSWG. The draft included a section entitled "An Open Systems Architecture" that was not yet complete. In this part is a copy of text provided by IDA to serve as introductory material for that section. The text provided by IDA defines the concepts of open system environments and architectures and discusses their potential benefits. It also briefly discusses factors that affect their "success."

Introductory text for Section 4 of the NGCR OSSWG draft document entitled *A Next Generation Computer Resources (NGCR) Open Systems Architecture*

Submitted by Karen Gordon (IDA)

4. AN OPEN SYSTEMS ARCHITECTURE

In P1003.0, the *Draft Guide to the POSIX Open Systems Environment*, several definitions related to the open system concept are offered. For example, an "open system" [P1003.0, Draft 14, p. 9, Sec. 2.2.2.23?] is defined to be

> A system that implements sufficient open specifications for interfaces, services, and supporting formats to enable properly engineered applications software:
>
> — to be ported with minimal changes across a wide range of systems
>
> — to interoperate with other applications on local and remote systems
>
> — to interact with users in a style that facilitates user portability.

"Open specifications" [P1003.0, Draft 14, p. 9, Sec. 2.2.2.21?] are defined to be

> Public specifications that are maintained by an open, public consensus process to accommodate new technologies over time and that are consistent with international standards.

An "Open System Environment (OSE)" [P1003.0, Draft 14, p. 9, Sec. 2.2.2.24] is defined to be

> The comprehensive set of interfaces, services, and supporting formats, plus user aspects for interoperability or for portability of applications, data, or people, as specified by information technology standards and profiles.

Taken together, these definitions define the open system concept in terms of (1) the potential benefits of open systems, (2) the "openness" of the specifications that define an open system, (3) the "openness" of the process through which the specifications are maintained, and (4) the notion of pulling together compatible standards and profiles into a set that defines a comprehensive computing environment.

NIST report PB91-201004, *Application Portability Profile (APP)—The U.S. Government's Open System Environment Profile OSE/1 Version 1.0* [NIST 92], offers a definition of "open system environment" that seems to reiterate the main points of all three P1003.0 open system definitions. In particular, the NIST report defines an open system

environment as a computing environment having the following three characteristics [NIST 92, p.3]:

- It is "based on an architectural framework that allows an extensible collection of capabilities to be defined."
- It is defined "in terms of nonproprietary specifications that are available to any vendor for use in developing commercial products."
- It evolves "through an open (public), consensus-based process for defining, specifying, and coordinating standards related to the computing environment."

The nonproprietary specifications referred to in the second characteristic are known as "open system standards." An "open system architecture" can be viewed as a computer system architecture developed in accordance with an open system environment. In other words, it is a high-level system design based on an architectural framework and defined in terms of open system standards.

Both the P1003.0 and the NIST definitions focus on the openness of the specifications as well as on the openness of the process through which the specifications are developed and evolved. The definitions differ in that the P1003.0 definitions emphasize potential benefits of open systems, and in that the NIST definition emphasizes the importance of working from an architectural framework that can help manage the development and evolution of the open system standards.

It should be noted that there seems to be general agreement on the fact that open system standards must be open, namely nonproprietary. However, there is some controversy over whether open system standards must necessarily be derived through an open, public, consensus-based process. The issue has to do with the fact that some company-produced specifications, such as MS-DOS, have become "public" and can and do offer the benefits claimed by open systems.

The roots of the OSE concept lie in experience gained in the world of data communication, in which standard protocols and services have for many years been recognized as a key to communication and interoperability among heterogeneous computer systems. The OSE concept can be viewed as a generalization of the ISO Open System Interconnection (OSI) concept—from the domain of communication to the domain of computing in general.

In the OSI effort, the seven-layer reference model serves as the "architectural framework." Numerous data communication protocol standards have been developed in

the context of the seven-layer reference model, and they serve as the "nonproprietary specifications." Several organizations, including ISO, ANSI, and the IEEE, serve as public forums for the "open, consensus-based process" of developing and evolving standards.

In ongoing OSE efforts, "architectural frameworks" are being developed in terms of categories of services. For example, in the POSIX OSE, five service areas (fundamental system services, communications services, information services, human-computer interaction services, and domain services) are used as the architectural framework in which to pursue standardization. In the NIST APP, seven service areas (operating system services, user interface services, programming services, data management services, data interchange services, graphics services, and network services) are being used as the architectural framework. In the CIM Reference Model for Computing, nine service areas (the seven NIST APP service areas plus security services and system management services) are being used as the architectural framework. Although the service areas do not provide as structured a framework for computing as the seven OSI layers do for communication, the service areas nevertheless do provide a useful categorization of work to be done.

With respect to "nonproprietary specifications," numerous standards have been developed, and many more are being developed, in each of the service areas noted above. With respect to the "open, consensus-based process" of pursuing standardization, the same organizations that sponsor OSI work are also sponsoring OSE work.

OSEs have the potential for making great strides toward the attainment of several important and interrelated goals:

- Application software portability. An OSE lays the foundation for application software portability by specifying standard application program interfaces to standard services. Application software that utilizes (only) the standard interfaces (and not proprietary enhancements) is portable across different implementations of the standard interfaces.
- Protection of application software investments. An OSE provides some protection against obsolescence caused by technological advancements in hardware. The idea is that the application program interfaces are designed to be independent of hardware technology. As technology evolves, new hardware platforms can host the same standard interfaces as their predecessors, and applications can be ported to the new platform. The burden of adjusting

to the new hardware platforms rests with the vendors of infrastructure software such as operating systems, and not with application developers.

- System and application interoperability. By defining standards for data communication and data exchange, an OSE supports the interoperability of heterogeneous computer systems and subsystems, as well as the interoperability of application software running on them.
- COTS acquisitions from multiple sources. An OSE enhances the prospects for being able to acquire COTS products from multiple sources. Since the standards are for interfaces and not implementations and since they are nonproprietary, multiple vendors can implement products that meet the standards.

The extent to which a specific OSE lives up to its potential depends upon the base of interest and support it achieves. That is, the success depends upon the number of customers demanding products conforming to the OSE, the number of vendors marketing products conforming to the OSE, and the number of organizations and individuals willing to support the consensus-based process for developing the standards underlying the OSE.

The success of an OSE also depends upon the far-sightedness of the standards developers. It is not always possible to design standard interfaces that can survive every revolutionary advance in technology. For example, some current data communication protocols may not be able to support the very high-speed data rates that are expected to become available in communication networks in this decade. At the same time, however, it is incumbent upon standards developers to consider scalability of the standards throughout the standardization process.

4.1 ...

PART 2

Summary of
POSIX Real-Time Application Environment Profiles

POSIX Working Group 1003.4 is the *real-time* working group. It is responsible for defining real-time extensions to the basic POSIX interfaces and also for defining real-time application environment profiles. At this time, it is developing specifications for four real-time profiles: (1) a minimal real-time system profile, (2) a real-time controller system profile, (3) a dedicated real-time system profile, and (4) a multi-purpose real-time system profile. In this part is a tabular summary of the four profiles, which was presented at the January 1992 NGCR OSSWG meeting. The tabular summary is preceded by background information, which explains the summary and why it was prepared and presented at the meeting.

Tabular Summary of POSIX Real-Time Application Environment Profiles

Presented by Karen Gordon (IDA), 14 January 1992

Background:

Before going to the tabular summary, it might be useful to review some background material:

NGCR OSSWG Representative Application Domains. As a part of its evaluation process, the NGCR OSSWG identified eight application domains to which the NGCR operating system interface standards should apply. These domains were called *representative application domains*, or RADs, and they were labeled as follows: (1) Ruby, interactive processing, (2) Opal, special-purpose processing, (3) Amethyst, reliable message processing, (4) Garnet, embedded processing (5) Topaz, high computation, (6) Emerald, mission-critical systems, (7) Diamond, networked processors, and (8) Sapphire, integrated subsystems. Brief textual descriptions of the RADs were written. In addition, each RAD was defined in terms of how important (on a scale of 0 to 10) each of fifteen different *classes* of requirements was to the RAD.

Application environment profiles. In the meantime, the open system standards community was formulating the concept of application environment profiles. A profile is a suite of standards, together with a selection of options within the standards. In the case of the POSIX standards, it was recognized that not all operating systems would need to implement all of the POSIX standards and all of their associated options. Instead, an operating system should implement a suitable profile, i.e., one whose target application environment matched that of the operating system.

The profiles being developed within the POSIX framework include (1) traditional multi-user interactive processing, (2) transaction processing, (3) high-performance computing, (4) multiprocessing, and (5) real-time processing. By specifying POSIX standardized profiles, the POSIX working groups are attempting to give vendors some targets for their implementations and to give users some well-known and widely-implemented profiles to call out in their procurements.

POSIX Real-Time Profiles. POSIX Working Group 1003.4, the group that is defining real-time extensions to the basic POSIX interfaces, is responsible for defining real-time profiles. After much deliberation, the group reached a consensus that there should be four

real-time profiles, representing four categories of real-time processing. The profiles are referred to as (1) minimal real-time system profile, (2) real-time controller system profile, (3) dedicated real-time system profile, and (4) multi-purpose real-time system profile. At the grossest level of detail, the first two profiles can be viewed as profiles for very small systems, with a file system in the second profile but not in the first. The last two profiles can be viewed as profiles for large systems, with a file system in the fourth profile but not in the third. The tabular summary describes the profiles in more detail.

Intrepretation of the tabular summary. The tabular summary shows which POSIX interfaces are included in each profile. POSIX.1 is the standard that specifies basic system interfaces, such as those for processes, files, pipes, and signals. POSIX.4 specifies real-time extensions, such as priority scheduling, memory locking, shared memory, and semaphores. POSIX.4a specifies threads interfaces, where a thread is a unit of control within a process. POSIX.2 and .2a specify shell and utilities interfaces. POSIX.8 specifies network-transparent file access interfaces. POSIX.12 specifies protocol-independent interprocess communication interfaces for networks.

The tabular summary should be interpreted as follows. The first column indicates which interfaces are included in the minimal real-time system profile. The second column, which describes the real-time controller system profile, indicates which *additional* features are included in the controller profile That is, the controller profile is equivalent to the minimal profile plus the interfaces listed in the second column. Likewise, the third column shows how to get from the controller profile to the dedicated profile. That is, the dedicated profile is equivalent to the controller profile plus the interfaces marked by a "+" sign and minus the interfaces marked by a "-" sign. (As noted earlier, file interfaces are present in the controller profile but not in the dedicated profile.) Finally, the multi-purpose profile includes *all* of the POSIX.1 interfaces (including the file system interfaces), *all* of the POSIX.4 interfaces (including the real-time file and mapped file interfaces), *all* of the POSIX.2 and .2a interfaces; and it *optionally* includes still other interfaces, as indicated in the summary.

Motivation for presenting the tabular summary. The NGCR OSSWG had decided at a previous meeting that members should prepare mappings of the NGCR OSSWG RADs to specific requirements (vs. requirements *classes*, as had been done during the evaluation process), and ultimately to specific POSIX interfaces. The idea was to begin to define some NGCR OSSWG profiles.

I had volunteered to cover two RADs—the embedded processing (Garnet) RAD and the interactive processing (Ruby) RAD. I thought that two POSIX profiles should be used as the baselines for these two RADs ; in particular, I thought that the POSIX minimal real-time system profile should serve as the baseline for the NGCR OSSWG embedded processing RAD and that the POSIX traditional multi-user interactive processing profile should serve as the baseline for the NGCR OSSWG interactive processing RAD. I believed that it was important for the NGCR OSSWG to use commercially accepted profiles as the baselines for NGCR OSSWG profiles, just as it was important to use commercially accepted standards (such as POSIX) as the baselines for the NGCR OSSWG interface standards. In the course of presenting the tabular summary, I suggested that the NGCR OSSWG start with the POSIX profiles, instead of trying to map RADs onto requirements and/or interfaces.

With respect to POSIX, I am a member of 1003.4, the real-time working group. I had participated in the development of the real-time profiles. Therefore, I decided to present a brief tabular summary of all the real-time profiles, so that the NGCR OSSWG members could get a feeling for where the 1003.4 Working Group was going. I thought that after reviewing these and other POSIX profiles, the NGCR OSSWG would be in a better position to determine the feasibility of relying on the POSIX profiles as baselines for the NGCR OSSWG profiles.

[Note: Over the last several months, RADs have faded into the background, and the NGCR OSSWG has placed increasing emphasis on the POSIX profiles. Recognition of the superior commercial viability of the POSIX profiles seems to be the primary reason behind the shift in emphasis.]

Summary of POSIX Real-Time Profiles

	Minimal	Controller	Dedicated	Multi-purpose
POSIX.1	single process basic I/O: open, close, read, write	+ file system + signals + pipe	+ multiple processes - file system	+ rest of POSIX.1
POSIX.4	semaphores memory locking shared memory timers message passing synchronized I/O	+ asynchronous I/O + real-time files + mapped files	+ real-time signals + priority scheduling + prioritized I/O - real-time files - mapped files	+ rest of POSIX.4
POSIX.4a	threads (all)			**optional** threads
Other				+ POSIX.2, .2a **optional** POSIX.8, POSIX.12, X Windows

PART 3

Contributions to

OSSWG Fault Tolerance Issues Paper

The NGCR OSSWG writes and maintains a series of papers on various issues of concern to the group. One of the papers (referred to above) is on the topic of fault tolerance. Fault tolerance is one the major requirements of mission-critical computing systems, but is not adequately addressed in the POSIX family of standards. The *OSSWG Fault Tolerance Issues Paper* is a collection of write-ups on several issues having to do with fault tolerance and how it might be addressed in the POSIX context. Some of the issues are technical in nature; others are more management oriented. The issues include the following: (1) level of interest within the POSIX community, (2) complexity of requirements, (3) conformance testing, (4) relationship to NGCR prototyping efforts, (5) inconsistent handling of faults/errors in the POSIX standards, (6) portability of services and interfaces, (7) timing of standardization, (8) relationship to security, (9) system-wide fault tolerance, (10) application-controlled policies and mechanisms, and (11) data replication. In this part are copies of write-ups provided by IDA on the last two of these issues.

"Application-controlled Fault Tolerance Policies and Mechanisms," a write-up of one of the issues covered in the NGCR OSSWG draft document entitled *OSSWG Fault Tolerance Issues Paper*

Submitted by Karen Gordon (IDA)

2.*n* ISSUE: APPLICATION-CONTROLLED FAULT TOLERANCE POLICIES AND MECHANISMS

Fault tolerance is defined by Nelson and Carroll as "the use of protective redundancy to permit continued correct operation of a system after the occurrence of specified faults" [Nelson and Carroll 87, p. 2]. The redundancy can be of various forms. Nelson and Carroll classify the forms into four categories: hardware (e.g., triple modular redundancy), information (e.g., error detecting/correcting codes, file replication), software (e.g., diagnostics software, n-version programming), and temporal (e.g., repeating operations, resending messages).

The management and implementation of redundancy clearly consumes resources—space (hardware) and/or time. Thus, there is an inherent tradeoff between fault tolerance and performance. In mission-critical systems, this tradeoff should be driven by mission needs. Mission needs can be conveyed to the operating system (the resource manager) by application software. In order for the application software (as coded by the developer of the application software) to be able to wisely make the tradeoff between fault tolerance and performance and to effectively control the management of resources, the following conditions should hold:

- Monitoring: The operating system should be capable of selectively monitoring the state of the system, as directed by the application software. This monitoring should include both (1) the usage of resources and (2) the occurrence of errors.
- Informing: The operating system should be capable of informing the application software of the state of the system, as requested by the application software.
- Control: The application software should be able to control the application of fault tolerance policies and mechanisms.

2.*n*.1 OPTIONS

1. The OSSWG could continue to work in existing POSIX working groups to ensure that application control of fault tolerance policies and mechanisms is adequately supported. Examples of ongoing work on the aspect of monitoring, informing, and enabling application control of resource usage include (1) timeouts on blocking services (here the "resource" is elapsed time) and (2) CPU time usage (here the resource is the CPU(s)). This concept of monitoring the usage of resources could be extended to other resources. For example, the usage of files could be recorded and used to improve the placement of files in a system (i.e., move files or make copies of files and place them at sites where they are most often accessed).

An example of application control of the tradeoff between fault tolerance and performance is the P1003.12 work in which messages with different qualities of service can be sent.

2. The OSSWG could work in a newly formed POSIX dot group on fault tolerance to add the necessary interfaces to the POSIX family of standards.

2.*n*.2 OSSWG POSITION

The OSSWG should pursue both options. Where feasible, the OSSWG should try to work within existing POSIX working groups. But, it seems that some of the OSSWG requirements (e.g., monitoring of resource usage, logging of errors, establishment of fault detection thresholds, creation of shadow files, checkpointing and frequency thereof) may not have a (near-term) home in any existing POSIX working group.

These requirements could perhaps be best addressed in a group focused on fault tolerance.

2.*n*.3 RESOLUTION

TBD/TBA

REFERENCES

[Nelson and Carroll 87] Nelson, Victor P. and Bill D. Carroll, Tutorial: Fault-Tolerant Computing, IEEE Computer Society Press, 1987.

"Data Relication," a write-up of one of the issues covered in the NGCR OSSWG draft document entitled *OSSWG Fault Tolerance Issues Paper*

Submitted by Karen Gordon (IDA)

2.*m* ISSUE: DATA REPLICATION

Information redundancy is one of the forms of redundancy that can be used to achieve fault tolerance. Information redundancy can be implemented at the low level through error detecting/correcting codes. At higher levels, it can be implemented through replication of data structures.

There are different replication policies that can be used to achieve different tradeoffs between fault tolerance and performance. In general, higher degrees of fault tolerance entail mechanisms such as locking or timestamps that lead to delays and reduce performance.

Atomic transactions are often used in conjunction with replication to manage the use of the replicated data structures. Again, there are different policies that can be used in the implementation of transactions.

OSSWG requirements that point toward the need for flexible data replication policies and mechanisms include 1/26 (Resilience), 1/27 (Network Partition), 6/15 (Shadow Files), 6/17&18 (Query & Modify File Attributes), 11/14 (Checkpointing), and 13/5 (Transaction Scheduling).

2.*m*.1 OPTIONS

1. The OSSWG could start participation in P1003.11, the POSIX working group that is defining a transaction processing AEP. The current draft is only 10 pages, and it seems to be in need of some support. The draft points to the work of other standards organizations, in particular, ISO, X/Open, and UNIX International. These organizations seem to have produced a lot of material on transactions. The OSSWG participant(s) in P1003.11 would have a lot of reading material, even though the draft itself is very short. The OSSWG participant(s) would need to ensure that the transaction interfaces being brought into the POSIX world are flexible enough to accommodate OSSWG requirements (i.e., to allow application control of the tradeoff between fault tolerance and performance).

2. The OSSWG could introduce interfaces for replicated files and checkpointing into an existing POSIX working group.

3. The OSSWG could introduce interfaces for replicated files and checkpointing into a new POSIX dot group on fault tolerance.

2.*m*.2 OSSWG POSITION

The OSSWG should pursue options 1 and 3. There is too much ongoing transaction work to ignore, so P1003.11 is important. Replicated files and checkpointing seem to have the most hope of progressing in a working group focused on fault tolerance.

2.*m*.3 RESOLUTION

TBD/TBA

PART 4

Review of

Selected NGCR Operating System Interface Requirements

The NGCR OSSWG's *Operational Concept Document* specifies requirements for the NGCR operating system interface standard. The NGCR OSSWG's *DELTA Document* gives the status of each requirement with respect to the POSIX standards. Both documents are meant to be reviewed and revised at regular time intervals. To this end, NGCR OSSWG members were asked to review the documents with respect to specific requirements. In this part is a copy of the review provided by IDA.

Proposed Revisions to *Operational Concept Document* and *DELTA Document*

Submitted by Karen Gordon (IDA)

Background:

The NGCR OSSWG's *Operational Concept Document*[1] contains the specification of requirements for the NGCR operating system interface standard. During the NGCR OSSWG's evaluation process, these requirements served as the criteria by which candidate operating system interfaces were evaluated. Then, when the evaluation was completed and POSIX was selected as the baseline upon which to build the NGCR operating system interface standard, the requirements listed in the *Operational Concept Document* assumed a new role. In particular, they now provide guidance to the NGCR OSSWG in its efforts to ensure that the POSIX standards evolve in directions useful to mission-critical computing.

The NGCR OSSWG's *DELTA Document*[2] gives the status of each requirement with respect to the POSIX standards. Section 3 of the *DELTA Document* summarizes the extent to which each requirement is met by the POSIX standards. Section 4 re-examines each unmet (or "unfulfilled") requirement, and it rates the feature represented by the requirement as being "a" (still required), "b" (highly desirable), "c" (deferrable), or "d" (in need of further evaluation). Section 6 proposes approaches for evolving the POSIX standards to meet the significant unfulfilled requirements (those that received ratings of "a" or "b" in Section 4), and it makes recommendations on which approaches to take.

Both the *Operational Concept Document* and the *DELTA Document* are living documents, meant to be reviewed and revised at regular time intervals. To this end, NGCR OSSWG members were asked to review the documents with respect to specific requirements. In this part is a copy of the review provided by IDA. It covers Requirement 1 of Service Class 2 (Architecture Dependent Interfaces), Requirements 2, 5, and 6 of Service Class 5 (Event and Error Interfaces), Requirement 6 of Service Class 7 (Generalized I/O Interfaces), and Requirement 1 of Service Class 9 (Process Management

---

[1] *Operational Concept Document for the Next-Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline*, NUSC Technical Document 6998, 1 April 1991.

[2] *DELTA Document for the Next Generation Computer Resources (NGCR) Operating Systems Interface Standard Baseline*, Version 2, 31 December 1991.

Interfaces). The review contains "marked-up" excerpts of the 1 April 1991 *Operational Concept Document* and the 31 December 1991 *DELTA Document*. It should be read as follows:

- Plain text is used for original text (i.e., text from the 1991 versions of the documents under review) that should remain unchanged.
- **Bold text** is used for new text that should be inserted.
- *[Italicized text in brackets]* is used for original text that should be deleted.
- **[Bold text in brackets]** is used for notes to the reader (e.g., to point out issues that need further discussion or decisions that were made at the September NGCR OSSWG meeting).

Service Class 2 (Architecture Dependent Interfaces), Requirement 1

Proposed Revisions to Operational Concept Document [None]

- 20.2.1 Non-NGCR System Interfaces

20.2.1.1 Definition
The OSIF shall support non-NGCR-based systems by providing a subset of its services to those systems. As a minimum this subset shall include:

- Download, initialize, start, and stop
- Ability to share resources, particularly peripheral devices
- Process-to-process message communication
- Ability to pass operational status information

20.2.1.2 Metric
20.2.1.3 Rationale
The Navy has a large investment in existing non-NGCR-based systems. These systems will continue to be in use for years to come and will need to interface to some degree with NGCR-based systems. Additionally, non-NGCR-based systems may need a method to gracefully transition to NGCR-based systems. The NGCR-systems will be required to accommodate interfacing to and evolution of the non-NGCR-based systems.

Proposed Revisions to DELTA Document

- 3.2 Architecture Dependent Interfaces

Architecture Dependent Interface (20.2.1) is met by:

P1003.1, paragraph 3.1 - 3.4
P1003.1, paragraph 6.1 - 6.5
P1003.4/**D12**, paragraph *[9.1 - 9.4]* **23.1 - 23.4**
P1003.4/**D12**, paragraph *[10.1 - 10.4]* **6.4 - 6.6**;

The OSIF shall support non-NGCR based systems by providing a subset of its services to those systems. The subset of service requests from non-NGCR based systems include **download,** initialize, start, resource sharing, process to process message communication, and ability to pass operational status information.

The non-NGCR system may issue service requests over non-NGCR or NGCR network interfaces. The NGCR network interfaces include FUTUREBUS+, SAFENET, and 1553B (See the OCD, Paragraph 20.8.1.1). The non-NGCR network interfaces include (but are not limited to) VME, MULTIBUS, TCP/IP, RS232, RS422 (See OCD, Paragraph 20.8.2.3).

POSIX does not provide explicit interfacing to non-NGCR networks. However, POSIX can support interfacing to non-NGCR networks given that the term "support" allows for hardware to be added to the non-NGCR network interface, and software to be added to both NGCR and non-NGCR systems. The application implementation of the additional hardware and software will allow the ability to service non-NGCR system service requests.

Requirements Coverage Summary

Requirement	Covered	POSIX Delta
2.1	Yes	None

- 4.2 Architecture Dependent Interfaces

There are no unmet requirements for service class 2.

Unfulfilled Requirements Rating

Requirement	Rating
2.1	-

- 6.2 Architecture Dependent Interfaces

There are no unmet requirements for service class 2.

Service Class 5 (Event and Error Interfaces), Requirements 2, 5, and 6

Proposed Revisions to Operational Concept Document

- 20.5.2 Event and Error Distribution

20.5.2.1 Definition
The OSIF shall provide for **specifying** the distribution of event and error information.
20.5.2.2 Metric
20.5.2.3 Rationale
This requirement is a necessary corollary to the Event and Error Receipt requirement (Refer to Section 20.5.1). The Fault Information Request requirement (Refer to Section 20.11.2) also applies to the error information distribution part of this requirement.

- 20.5.5 Enable/Disable Interrupts

20.5.5.1 Definition
The OSIF shall provide the ability to enable and disable interrupts.
20.5.5.2 Metric
20.5.5.3 Rationale
This requirement provides for interrupts as a whole to be turned on and off. The mask/unmask interrupts requirement, on the other hand, provides for individual interrupts to be made known/unknown.

- 20.5.6 Mask/Unmask Interrupts

20.5.6.1 Definition
The OSIF shall provide the ability to mask and unmask *[events]* **interrupts**.
20.5.6.2 Metric
20.5.6.3 Rationale
A system requires this capability during such activities as interrupt processing to lock out interrupts of a lower class from occurring or to mask out the interrupts from particular sources.

Proposed Revisions to DELTA Document

- 3.5 Event and Error Interfaces

...

OSSWG requirements 5.1 and 5.2 are partially covered by POSIX. While the event interfaces exist, and error interfaces are provided for individual processes, there are no error coordination or distribution interfaces.

[Note that this paragraph and the following one have been revised in accordance with the outcome of the September NGCR OSSWG meeting.] It is anticipated that 1003.4b will provide the capability for an application to specify that, upon occurrence of a designated interrupt, a signal is to be sent to a designated process, or a designated user-written ISR is to be executed (or both). This interrupt control capability, in conjunction with 1003.1/1003.4/1003.4a signals, would provide some coverage of

requirement 5.2 (distribution of event and error information). In particular, the interrupt control mechanism could be used for the distribution of information on events and errors resulting in hardware interrupts (such as hardware device errors). However, this distribution mechanism would not be applicable to certain operating system errors (such as those in which kernel data structures become faulty).

Another possible deficiency in the coverage of requirement 5.2 is the fact that functions return indication of only a single error, instead of all errors that occur during function processing.

...

For requirements 5.5 **(enable/disable all interrupts)** and 5.6 **(mask/unmask selected interrupts)**, POSIX has considered direct control of interrupts to be "out of scope" and hence not provided. However, 1003.4b is now broaching the subject and suggesting that interrupts be handled in a way similar to signals.

It is anticipated that 1003.4b, in its interrupt control chapter, will provide functions for enabling and disabling interrupts. However, the 1003.4 Working Group still considers masking/unmasking of interrupts to be too hardware dependent to be standardized. (Note that the proposed 1003.4b approach enables interrupts to be connected to signals and signals to be masked/unmasked.)

Requirements Coverage Summary

Requirement	Covered	POSIX Delta	
5.1	Partially	Insertion	
5.2	Partially	Insertion	**[still only partially covered, according to September meeting]**
5.3	Partially	Insertion	
5.4	No	Insertion	
5.5	**Yes**	**None**	**[with 1003.4b's interrupt control]**
5	Partially	Insertion	

- [illegible]vent and Error Interfaces

...

However, some philosophical views and assumptions of the POSIX community, differ considerably from that demonstrated by the OSSWG conceptual model.

Examples are access to interrupts and error logging. Both were cited as "out of scope" by the POSIX community.

1003.4b is currently developing interrupt control interfaces which will presumably fulfill Requirement 5.5 and contribute to the fulfillment of Requirement 5.2. Due to hardware dependencies, it may not be appropriate to attempt to standardize interfaces for masking/unmasking interrupts.

Unfulfilled Requirements Rating

Requirement	Rating	
5.1	-/c	
5.2	a	**[still "a", according to September meeting]**
5.3	a	
5.4	a	
5.5	*[a]*	
	-	**[assuming 1003.4b interrupt control covers the requirement]**
5.6	*[a]*	
	c	**[not ready for standardization???]**

- 6.5 Event and Error Interfaces

...

We recommend satisfying 6.5.5, Enable/Disable Interrupts, and 6.5.6, Mask/Unmask Interrupts in 1003.4b where this work has already been undertaken. **Mask/Unmask Interrupts may not be provided by 1003.4b, because of hardware dependencies. (Classification as a significant unfilled requirement should be reconsidered, as indicated in Section 4.5 of this Delta Document.)** Requirement 6.5.1, Event and Error Receipt, is deferred as far as errors are concerned because this is not an API requirement.

...

6.5.2 Event and Error Distribution

Requirement: The OSIF shall provide for **specifying** the distribution of event and error information. This is a necessary OSIF requirement.

Description of Deltas: POSIX provides for the distribution of events through Signals (paragraph 3.3, IEEE Std 1003.1-1990). Table 3-1 (IEEE Std 1003.1-1990) lists the signals that all POSIX implementations must support and Table 3-2 (IEEE Std 1003.1-1990) lists those signals that a system implementing job control must support. However, "an implementation may define additional signals that may occur in the system" (paragraph 3.3.1.1, IEEE Std 1003.1-1990). **The Signals interface is enhanced in 1003.4/D12, Section 3.3, and it is extended to threads in 1003.4a/D6, Section 8.**

POSIX provides for the distribution of errors to the requesters of individual functions. Each function specifies which errors all POSIX implementations must detect and which are optional. Paragraph 2.4 (IEEE Std 1003.1-1990) lists the possible errors. However, "implementations may support additional errors not included in this clause, may generate errors included in this clause under circumstances other than those described in this clause, or may contain extensions or limitations that prevent some errors from occurring" (paragraph 2.4, IEEE Std 1003.1-1990). "If more than one error occurs in processing a function call, this part of ISO/IEC 9945 does not define in what order the errors are detected; therefore, any one of the possible errors may be returned" (paragraph 2.4, IEEE Std 1003.1-1990). **[Can this approach be tolerated?]** In addition, realtime extensions to POSIX in 1003.4b is pursuing handling of interrupts. **In 1003.4b**

(expected in draft 4 or 5), the occurrence of an interrupt can be made to trigger the sending of a signal to a designated process, or the execution of a user-written ISR (or both).

The OSIF requires that all possible errors be available, not just one of those possible. **[Again, can this be tolerated?]** It also requires that there be a means for coordinating the distribution of errors, as for example to a single process responsible for error analysis. **The 1003.4b interrupt control interface enables distribution of certain errors, namely those resulting in hardware interrupts. [Note that the preceding line was revised as a result of the September meeting.]**

Recommendation: We recommend that the OSSWG change the wording of this requirement to be more consistent with similar requirement 6.11.1. Possible new wording is: The OSIF shall provide for **specifying** the distribution of event and error information.

We recommend that OSSWG continue to support the work to make interrupts available through 1003.4b interfaces. **Interrupt control is expected to become available in Draft 4 or 5 of 1003.4b.**

[And, to completely satisfy this requirement, we recommend that OSSWG pursue a POSIX PAR to add system fault and error management to the work of 1003.7 or to begin a new system fault and error management group.] **[Align with other Service Class 5 and Service Class 11 recommendations.]**

...

6.5.5 Enable/Disable Interrupts

Requirement: The OSIF shall provide the ability to enable and disable interrupts. This is a necessary OSIF requirement.

Description of Deltas: POSIX does not currently provide the capability to handle interrupts. However, realtime extensions to POSIX in 1003.4b is currently pursuing handling of interrupts. **Interrupt control is expected to be included in draft 4 or 5 of 1003.4b.**

Recommendation: We recommend that the OSSWG continue to support the work in 1003.4b to resolve the delta for this requirement.

6.5.6 Mask/Unmask Interrupts **[Remove from Chapter 6???]**

Requirement: The OSIF shall provide the ability to mask and unmask events. This is a necessary OSIF requirement.

Description of Deltas: Within the limits discussed under requirement 6.5.2—i.e., POSIX does not provide for the collection and coordination of all events and errors—POSIX provides the ability to mask and unmask events through its signal processing (paragraph 3.3.1.2, IEEE Std 1003.1-1990). Therefore, complete resolution of the deltas for this requirement depend upon the resolution of requirement 6.5.2.

POSIX does not currently provide the capability to handle interrupts. However, realtime extensions to POSIX in 1003.4b is currently pursuing handling of interrupts. **Assuming that the requirement is to mask/unmask interrupts, the issue becomes**

whether it is possible to standardize this functionality. Hardware dependencies may make it inappropriate to try to standardize.

Recommendation: We recommend that the OSSWG change the wording of this requirement to be consistent with its title and with requirement 6.5.5: The OSIF shall provide the ability to mask and unmask interrupts. In this regard, the second paragraph of the description applies and we recommend that the OSSWG *[continue to support the work in 1003.4b to resolve the delta for this requirement]* **view this requirement as inappropriate for standardization.**

Service Class 7 (Generalized I/O Interfaces), Requirement 6

Proposed Revisions to Operational Concept Document [None]

- 20.7.6 Device Event Notification

Refer to requirements within Section 20.5, Event and Error Interfaces.

Proposed Revisions to DELTA Document [None]

- 3.7 Generalized I/O Interfaces

...

Refer to Section 3.5, Event and Error Interfaces, Device Event Notification (7.6).

Requirements Coverage Summary

Requirement	Covered	POSIX Delta
7.1	No	Insertion
7.2	Yes	None
7.3	Yes	None
7.4	Yes	None
7.5	Yes	None
7.6	Partially	Insertion
7.7	Partially	Modification
7.8	Yes	None
7.9	Yes	None
7.10	No	Insertion
7.11	No	Insertion

- 4.7 Generalized I/O Interfaces

...

Unfulfilled Requirements Rating

Requirement	Rating
7.1	a
7.2	-
7.3	-
7.4	-
7.5	-
7.6	-
7.7	a
7.8	-
7.9	-
7.10	a
7.11	d

- 6.7 Generalized I/O Interfaces

...

6.7.6 Device Event Notification

Refer to Section 3.5, Event and Error Interfaces

Service Class 9 (Process Management Interfaces), Requirement 1

Proposed Revisions to Operational Concept Document

- 20.9.1 Create Process

20.9.1.1 Definition
The OSIF shall *[provide]* **support** the ability to create processes with specified attributes.
20.9.1.2 Metric
20.9.1.3 Rationale
Processes *[and their environments]* need to be created, **and appropriate values need to be assigned to their attributes,** prior to their execution. Attributes may include such things as process name, process priority, stack size, scheduling attributes, memory allocation, etc.

Proposed Revisions to DELTA Document

- 3.9 Process Management Interfaces

...

The requirements for Create Process (9.1), Interprocess Communication (9.8), Examine Process Attributes (9.9), Modify Process Attributes (9.10), Process Identification (9.12), and Program Management (9.14) are directly met for Pthreads by P1003.4a plus the interprocess communication facilities of P1003.4. The requirements for Interprocess Communication (9.8), Examine Process Attributes (9.9), Modify Process Attributes(9.10), Process Identification (9.12), and Program Management (9.14) are directly met for POSIX processes by P1003.1 process interfaces plus P1003.4 process attributes and interprocess communication facilities.

The Create Process (9.1) requirement is met for processes by the fork and exec functions of P1003.1, the spawn interface of P1003.4b, the scheduling interface of P1003.4, plus the communication and synchronization interfaces of P1003.4.

...

The requirement for Create Process (9.1) is not adequately covered for POSIX processes because no attributes may be specified at creation time. P1003.4 associates certain attributes (priority, scheduler, etc.) with POSIX processes, but does not provide an interface allowing these attributes to be assigned at process creation time. **With "shall provide" changed to "shall support", the fact that process creation entails multiple steps (i.e., fork/exec or spawn, setting of attributes, and synchronization to hold back starting) is acceptable.**

Also, the POSIX process creation mechanism requires use of several P1003.1 interfaces used in combination; i.e. fork(), exec(), and the file system namespace. This is a rather awkward and expensive mechanism, and does not adequately address the creation of processes from memory resident code and data segments. **The spawn interface of P1003.4b alleviates the fork/exec problems.**

Requirements Coverage Summary

Requirement	Covered	POSIX Delta
9.1(Pthreads)	Yes	None
9.1(Process)	*[Partially]*	*[Modification]*
	Yes???	**None??? [with "shall support" and Spawn]**
9.2	Partially	Modification
9.3	No	Insertion
9.4	No	Insertion
9.5	Partially	Insertion
9.6	Partially	Insertion
9.7	Partially	Insertion
9.8	Yes	None
9.9	Yes	None
9.10	Yes	None
9.11	No	Insertion
9.12	Partially	Modification
9.13	No	Insertion
9.14	Yes	None

- 4.9 Process Management Interfaces

Unfulfilled requirement: Create Process (9.1)

This requirement would be classified as *[(b) Highly Desirable]* **(a) Required.** P1003.4 currently does not allow the attributes of a POSIX process (as defined in P1003.4, e.g. priority, scheduler) to be set concurrently with process creation (via "fork" and/or "exec"). These must be set via separate interfaces, which creates a window in time during which a process may execute with anonymous or undefined attributes. **Attributes are inherited and then can be changed. (This is acceptable when "shall provide" is changed to "shall support".)** P1003.4a provides such a mechanism for Pthreads, namely the thread creation attribute object. This creates an inconsistency in model between the Pthread and POSIX process creation. Since real time systems seldom (or never) do extensive process creation operations during time-critical operational scenarios, the window effects are not likely to severely perturb system operations; but the inconsistency should be eliminated by adapting the concept of creation attributes to P1003.1 processes, perhaps as an optional alternative form of "fork."

Additionally, the P1003.1 semantics of process creation must be modified or clarified to ensure that: a) a combined fork()/exec() operation need not incur the overhead of duplicating the running process; and b) the pathname passed to exec() need not necessarily imply loading of memory from a file (but may, for example, specify process code already memory resident). **The P1003.4b Spawn interface is an acceptable resolution of these concerns.**

...

Unfulfilled Requirements Rating

Requirement	Rating	
9.1	b	
	-	**[because of "shall provide" being changed to "shall support", and because of P1003.4b's Spawn]???**
9.2	a	
9.3	d	
9.4	d	
9.5	d	
9.6	d	
9.7	d	
9.8	-	
9.9	-	
9.10	-	
9.11	a	
9.12	a	
9.13	a	
9.14	-	

- 6.9 Process Management Interfaces

6.9.1 Create Process

Requirement 9.1: Create Process

Description of Delta: POSIX combines the create process/thread and start process/thread into one operation. **(This is acceptable when "shall provide" is changed to "shall support".)** OSSWG requires the process to be created so that it can be readied for the scheduler queue this is a two step operation. This more closely matches the Ada definition which separates the creation and start operations.

POSIX process attributes cannot be defined at process creation when using the FORK EXEC. **Some attributes (e.g., scheduling attributes) cannot be defined when using Spawn. (This is also acceptable when the requirement is "shall support".)**

Resolution: The delta can be minimized to more closely resemble the OSSWG definition if POSIX create thread would also use a conditional wait immediately following the create operation.

POSIX P1003.4b is exploring the possibility of introducing a process creation without FORK which allows (some) process attributes to be defined at Process Creation. **P1003.4b now includes a Spawn interface, which serves as an alternative to fork/exec.**

Recommendation: Modify the OSSWG requirement by changing shall provide to shall support and track P1003.4b to see if a Process Creation without Fork gets defined. **(Assuming that the recommendation to change "shall provide" to "shall support" is accepted, and that Spawn survives in P1003.4b, Requirement 9.1 is met.)**

...

LIST OF ACRONYMS

ANSI	American National Standards Institute
API	Application Program Interface
APP	Application Portability Profile
CIM	Corporate Information Management
COTS	Commercial off the Shelf
CPU	Central Processing Unit
I/O	Input/Output
IDA	Institute for Defense Analyses
IEEE	Institute for Electronics and Electrical Engineers, Inc.
ISO	International Organization for Standardization
ISR	Interrupt Service Routine
NGCR	Next Generation Computer Resources
NIST	National Institute for Standards and Technology
NUSC	Naval Underwater Systems Center
OSE	Open System Environment
OSI	Open System Interconnection
OSIF	Operating System Interface
OSSWG	Operating Systems Standards Working Group
POSIX	Portable Operating System Interface for Computer Environments
RAD	Representative Application Domain
TBA	To Be Announced
TBD	To Be Determined
TCP/IP	Transmission Control Protocol/Internet Protocol